RL-TR-94-165
In-House Report
October 1994

ENCODING USER-DEFINED PROBLEMS WITH SIPE-2

Karen M. Alguire

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-94-165 has been reviewed and is approved for publication.

APPROVED: Samuel A. DiNitto Jr.

SAMUEL A. DINITTO, JR., Chief
Software Technology Division

FOR THE COMMANDER: John A. Graniero

JOHN A. GRANIERO
Chief Scientist
Command, Control, and Communications Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1994	In-House Jun 93 - Mar 94

4. TITLE AND SUBTITLE
ENCODING USER-DEFINED PROBLEMS WITH SIPE-2

5. FUNDING NUMBERS
PE - 62702F
PR - 5581
TA - 27
WU - 61

6. AUTHOR(S)
Karen M. Alguire

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3CA)
525 Brooks Road
Griffiss AFB NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
RL-TR-94-165

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3CA)
525 Brooks Road
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Karen M. Alguire/C3CA (315) 330-4833

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report describes work related to some experiments performed at Rome Laboratory with generative planners. The main objective of the project was to provide an understanding of generative planners and familiarization with two different generative planners, O-Plan2 and SIPE-2. The approach was to become proficient in the use of the generative planning systems to define and solve new problems representative of basic planning capability requirements. The Missionaries and Cannibals (MC) puzzle was selected as a starting point because of its simplicity and it deals with simple numerical calculations, an aspect we felt was relevant to application interests. This report focuses on the work with SIPE-2, specifically, our experience encoding the MC puzzle. The process of encoding problems in general is also addressed for providing guidance in defining and solving user-defined problems with SIPE-2.

14. SUBJECT TERMS
Generative Planners, Artificial Intelligence, Nonlinear Planning Systems, Hierarchical Planning Systems

15. NUMBER OF PAGES
48

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	U/L

Contents

1 Introduction

- 1.1 Overview
- 1.2 Structure of Report

2 Approach Embodied in SIPE-2

- 2.1 SIPE-2's View of the Planning Problem
- 2.2 SIPE-2 Planning Features
  - 2.2.1 Hierarchical Planning
  - 2.2.2 Nonlinear Planning
  - 2.2.3 Deductive Causal Theory
  - 2.2.4 Planning Variables and Constraints
  - 2.2.5 Plan Execution and Replanning
  - 2.2.6 Domain-Independence
- 2.3 How SIPE-2 Produces New Planning Levels

3 Encoding New Problems in SIPE-2

- 3.1 Definition of the Sort Hierarchy
- 3.2 Definition of the Initial World Predicates
- 3.3 Definition of the Operators
- 3.4 Definition of the Problem(s)

4 The Missionaries and Cannibals (MC) Puzzle

5 Modeling the MC Puzzle in SIPE-2

- 5.1 The MC Puzzle Objects
- 5.2 The MC Puzzle Predicates
- 5.3 The MC Puzzle Operators
  - 5.3.1 Operators Controlling Movement
  - 5.3.2 Operators Performing Movement
- 5.4 Domain Theory in the MC Puzzle

5.5 Problem Definition for the MC Puzzle

6 Generating SIPE-2 Plans for the MC Puzzle

7 Lessons Learned

7.1 The Use of Numerical Variables

7.2 The Encoding of Operators with Multiple Effects

7.3 The Phantomization of Goals Through Variable Instantiation

8 Conclusion

9 Acknowledgments

References

Appendix Implementation of the Missionaries and Cannibals Puzzle in SIPE-2

1 Introduction

1.1 Overview

This report describes work related to some experiments performed at Rome Laboratory with generative planners. This project was a collaborative effort between Dr. Carla Ludlow[1], and Karen Alguire and Albert Frantz[2] from Rome Laboratory. The main objective of the project was to provide an understanding of generative planners and familiarization with two different generative planners, O-Plan2 ([O-Plan2 93], [Currie & Tate 91]) and SIPE-2 ([Wilkins 93], [Wilkins 88]). A secondary objective of the project, mainly a subproduct of the experimentation with both planners, was the comparison between O-Plan2 and SIPE-2 [Ludlow 94]. The system versions used for the project were O-Plan2 version 2.1 and SIPE-2 version 4.3 which was later upgraded to version 4.4.

Our approach to accomplishing the project objectives was to become proficient in the use of the generative planning systems to define and solve new problems representative of basic planning capability requirements. We decided the best way to assimilate knowledge of the planning systems was to first go through the demonstration examples provided with the systems and second, to focus on particular problems to encode starting with simple classical planning problems in the Artificial Intelligence (AI) community. The Missionaries and Cannibals (MC) puzzle was selected as a starting point because of its simplicity and it deals with simple numerical calculations. This was an aspect we felt was relevant to application interests.

The implementation of the MC puzzle as described in this report is the product of several iterations. We encountered some problems implementing the MC puzzle in SIPE-2 related to the use of numerical variables, the encoding of operators with multiple effects, and goal phantomization. These issues will be discussed in more detail in this report. We also realized that we would not be able to solve the MC puzzle at all with the current version of O-Plan2 as it does not handle numerical calculations. Since system evaluation was one of our objectives, we excluded the possibility of implementing our own arithmetic symbolically in O-Plan2 (e.g., using successor and predecessor functions). Issues related to the implementation of the MC puzzle in both SIPE-2 and O-Plan2 are described in the paper "Looking at O-Plan2 and SIPE-2 Through the Missionaries and Cannibals" [Ludlow & Alguire 94].

This report focuses on the work with SIPE-2, specifically, our experience encoding the MC puzzle. The process of encoding problems in general is also addressed for providing guidance in defining and solving user-defined problems with SIPE-2.

1.2 Structure of Report

Section 2 gives an overview of the SIPE-2 planning system developed by SRI International and highlights some of the key program features. This includes a discussion of SIPE-2's view of the classical planning problem and system features such as hierarchical planning, nonlinear planning, deductive causal theory, planning variables and

[1] Dr. Ludlow's contribution to this project was sponsored under Rome Laboratory contract F30602-93-C-0017.

[2] Albert Frantz was involved in the project for only the first two months.

constraints, plan execution and replanning, and domain-independence. The algorithm SIPE-2 uses to produce new planning levels is also presented.

Section 3 discusses the encoding of new problems with SIPE-2. This includes what SIPE-2 expects as input and how one might go about defining the domain information.

Section 4 presents the problem statement, legal moves, and constraints of the MC Puzzle.

Section 5 describes how the MC puzzle was modeled in SIPE-2. This includes domain information such as the object hierarchy, predicates, deductive rules, and operators defined in the world.

Section 6 explains how SIPE-2 uses the MC puzzle input to generate plans for solving the problem.

Section 7 captures some lessons learned through our MC encoding experience. Several aspects are discussed including the use of numericals, the encoding of operators with multiple effects, and goal phantomization through variable instantiation.

2 Approach Embodied in SIPE-2

SIPE-2 (System for Interactive Planning and Execution) is a hierarchical, domain-independent, nonlinear AI generative planning system developed by SRI International AI Center. SIPE-2 provides a formalism for describing actions and utilizes the knowledge encoded in this formalism, together with its heuristics for handling the combinatorics of the problem, to generate plans to achieve given goals in diverse problem domains [Wilkins 93]. It has mechanisms for reasoning about context-dependent effects, planning variables, variable constraints, and resources. Temporal reasoning in SIPE-2 treats time as a consumable resource whose consumption over parallel tasks is nonadditive. Recently, SIPE-2 has been extended to cooperate with General Electric's temporal reasoning system, Tachyon [Arthur *et al.* 93]. SIPE-2 has some plan execution and replanning capabilities. It is a component of SRI's CYPRESS system [Wilkins *et al.* 94] which supports planning capabilities including action specification, generative planning, reasoning with uncertainty, reactive plan execution, and dynamic replanning. Some of the key program features of SIPE-2 mentioned here will be highlighted in Section 2.2.

2.1 SIPE-2's View of the Planning Problem

In the classical planning problem, the world is represented by states. States are snapshots of the world at a particular point in time. Performing actions causes the world to move from one state to the next. In the problem specification, it is necessary to define a description of the initial state of the world, a set of allowable actions, and a set of goals. The remaining task of the planner is to find a sequence of actions that will transform the world from the initial state to the goal state.

SIPE-2's view of the planning problem closely resembles the classical planning problem. Figure 1 depicts the approach embodied in SIPE-2.

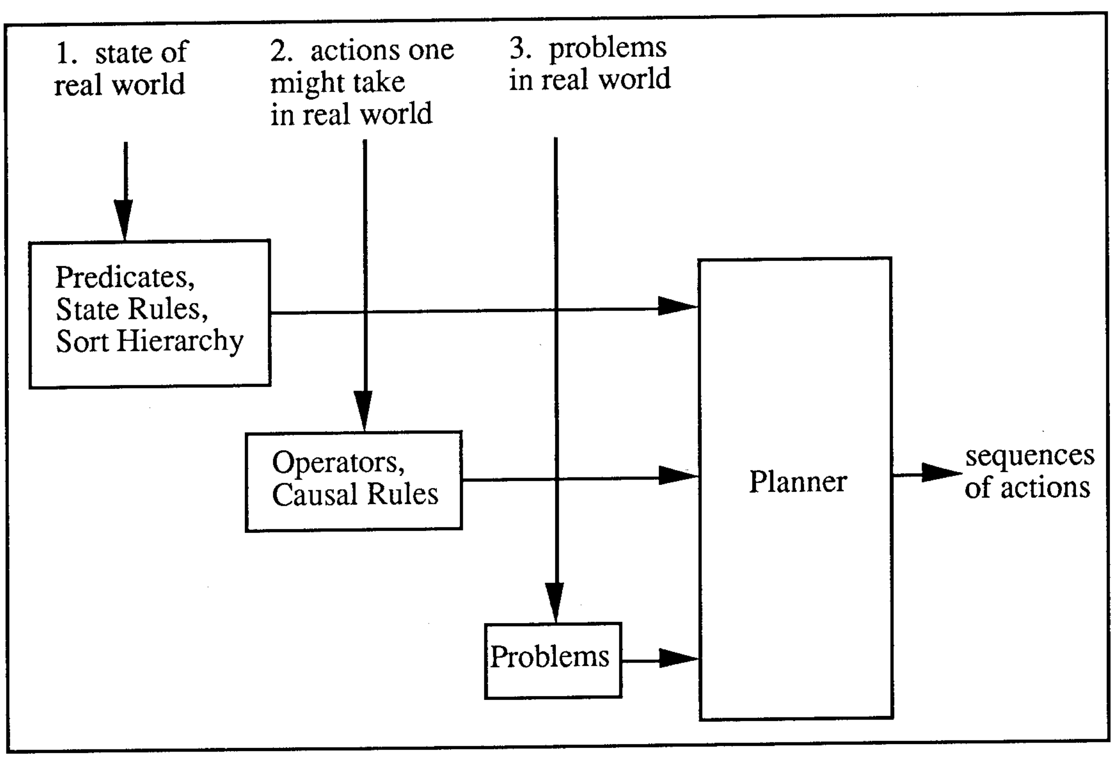

Figure 1. SIPE-2's View of the Planning Problem [Wilkins 93]

The inputs of the SIPE-2 system consist of a description of the initial state of the world, a description of a set of actions, a problem descriptor, and a set of rules describing the domain. The initial state of the world is represented by a sort hierarchy, a world model, and a set of deductive rules. The sort hierarchy represents invariant properties of perpetual objects and their structure in terms of classes with inheritance properties. The world model is a set of predicates that hold over objects in the sort hierarchy. Deductive rules, such as state and causal rules, are a mechanism for allowing effects of an event to be automatically deduced. The system's representations of actions or abstractions that may be performed in the domain are referred to as operators. While the inputs to the planner attempt to model the "real world," current AI techniques cannot handle the full complexity of the everyday world. In actuality, the problem being solved is always in a limited environment that is usually an abstraction of the real world.

The output of SIPE-2 is a sequence of primitive actions to be carried out by whatever agents are accepting instructions from the planner. The system automatically, or with interactive control, generates plans (i.e., sequences of known actions) to solve the given problem and supports replanning after unexpected occurrences during execution. A tree of alternative plans is produced by the planner. The planner's search for a solution consists of a simple depth-first search strategy until an acceptable plan has been produced. It is important to note that the output of the planner should be considered correct only to the extent that the encoded representation correctly reflects the real environment.

2.2 SIPE-2 Planning Features

2.2.1 Hierarchical Planning

SIPE-2 is a hierarchical planner, meaning that it supports different levels of abstraction, both in the planning process and in the description of the domain. In this context, a level of abstraction is distinguished by granularity, or fineness of detail. Hierarchical planning enables the planner to defer consideration of the details of a problem. The method is first to construct an abstract plan leaving details unspecified, and then to refine these components into more detailed subplans until enough of the plan has been filled in to ensure success. This approach is advantageous because the planner can attempt to develop a plan at a level that is not as computationally overwhelming. A simplistic example would be the strategy one might use to plan a vacation. At the highest level, it is enough to know in general terms when you want to go, where you want to go and what you want to do, e.g., you want to go to Florida around the first week of December, and you want to have fun! The next level of abstraction might be determining the details for the when, where and what, e.g., fly into Ft. Myers December 1st in the morning, spend eight days relaxing and sight-seeing on Marco Island, return home December 10th in the evening. The next level of abstraction might consist of booking the flight reservations, making hotel reservations, arranging for a rental car, and browsing through some Marco Island Tour Guide books and maps for recreational ideas. At the lowest level of planning, you will have a detailed vacation plan.

2.2.2 Nonlinear Planning

SIPE-2 is a nonlinear planner. Nonlinear planning is a technique of deferring decisions on the ordering of plan elements until such decisions are forced. Thus, instead of ordering the execution of the steps of a plan, the steps are left unordered until some conflict is detected, at which time an ordering constraint is imposed to remove the conflict. A plan is nonlinear if it contains actions that are unordered with respect to each other, i.e., actions for which the planner has not yet determined an order and which may be in parallel [Wilkins 93]. A planner's ability to avoid committing to a particular order of action until further information regarding order is accumulated cuts down the exponential search of all possible plan orderings. One problem that arises from nonlinear planning is the detection of interaction between parallel actions. The planner must reason about how actions that may take place concurrently interact with each other. Often, two actions may interfere with each other if they are executed simultaneously. Nonlinear planners must recognize and correct these situations.

2.2.3 Deductive Causal Theory

A powerful feature of SIPE-2 is the capability to encode a causal theory to represent and reason about the effects of actions in different world states. By allowing the separation of knowledge about actions from knowledge about causality, SIPE-2 provides a mechanism for implementing a causal theory of the domain [Wilkins 88]. Deductive rules are used to allow effects of an event to be deduced without being mentioned explicitly in operators. Thus, the deduction of context-dependent effects is permitted. Since conditional effects are deduced, operators can be applied to a wider range of situations. In complex domains, this can potentially reduce the number of operators required by a strict STRIPS assumption exponentially. By the STRIPS assumption, you would need a distinct operator for every different context in which the operator might be used.

2.2.4 Planning Variables and Constraints

The use of planning variables allows certain objects or entities within a plan to be left unspecified during the planning process. While the use of variables can cause complications (e.g., determining when a predicate containing a variable is true), their use can eliminate a huge search space since later information may determine the correct instantiation for a variable without having to produce and check a plan for every possibility. The search through possible variable instantiations can be further reduced if the planner can narrow the set of possible instantiations by allowing the user to specify constraints on the variables.

2.2.5 Plan Execution and Replanning

SIPE-2 supports plan execution and replanning. During execution of a plan, a person or computer system monitoring the execution can specify what actions have been performed and what changes have occurred in the domain being modeled. At any point during execution, the execution monitor will accept two types of information about the domain: an arbitrary predicate whose arguments are ground instances that is now true, false, or unknown; and a local variable name that is now unknown. Once the description of the unexpected situation has been accumulated, the execution monitor calls the problem recognizer, which returns a list of all problems it detects in the plan. The general replanner is then given the list of problems found by the problem recognizer and tries certain replanning actions in various cases.

2.2.6 Domain-Independence

SIPE-2 is a domain-independent planning system. This means that domain knowledge is encoded in the formalization of the problem to be solved in whatever domain the user chooses. Domain-independent planners must provide the user with a knowledge representation language for encoding domain-specific knowledge. SIPE-2 provides a powerful formalism for representing domains and automatically generating plans in them.

The performance of the planner depends on the quality of the domain knowledge encoded by the user. The SIPE-2 system has been applied to the classic AI blocks world, planning the actions of a mobile robot, planning the movement of aircraft on a carrier deck, travel planning, construction tasks, and transportation planning for Joint Forces course-of-action development. Because of its generic technology, SIPE-2 has the potential to affect a large variety of problems in fields as diverse as manufacturing, construction, and the military.

2.3 How SIPE-2 Produces New Planning Levels

Intelligent control of the search process has not been addressed in the development of SIPE-2 partly because searching algorithms and heuristics need to be domain dependent [Wilkins 88]. SIPE-2's automatic search mechanism searches through the space of partial plans depth-first by planning level, with chronological backtracking. This straightforward search does not perform well on large, complex problems if a solution is hard to find. However, SIPE-2 has been designed and built to support interactive planning, and SIPE-2 operators can be used to control search through the use of preconditions, conditions, or metalevel operators.

The search is responsible for balancing time spent checking critics and the global constraint network with time spent planning. The default is to check the critics once every planning level and before a final plan is accepted. The frequency of critic application can be set by the user through interface command menus.

Figure 2 depicts how the search algorithm produces a new planning level from the previous level.

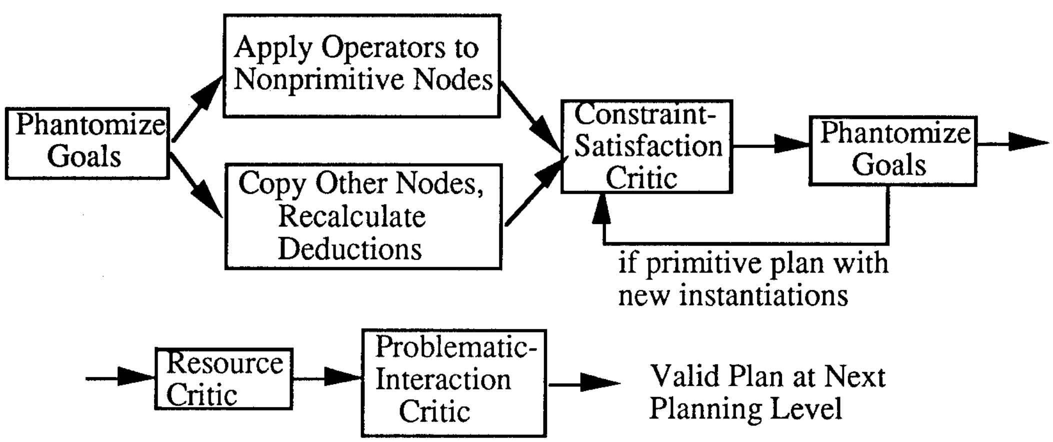

Figure 2. SIPE-2's Algorithm for Producing a New Planning Level [Wilkins 93]

The search begins by trying to phantomize any goal in the original problem (i.e., make a goal true without applying an operator). It then applies an operator to each node in the plan that requires further specialization (i.e., to each nonprimitive goal, process, or choiceprocess node). During this application of operators, any intervening nodes (e.g., precondition nodes, primitive nodes) adjacent to the nodes are copied to the new plan. The deduced effects of the copied nodes are recalculated, since they may change in the different context specified by the more detailed plan. The search checks for problems any of these operations may cause, and may change phantom nodes back to goals, as well as rejecting operator applications that falsify preconditions already present in the remainder of the plan.

After the above operations have created a plan at the new planning level, the critics are called on the new detailed plan. They add any constraints necessary to ensure that the global constraint network can be satisfied, then try to phantomize goal nodes, then check for resource conflicts, and finally check for problematic interactions among unordered actions. Both of the last two operations may add ordering constraints to fix conflicts or problems, which in turn may lead to nodes being added or removed from the plan.

SIPE-2 continues in this fashion until all goals have been successfully expanded which gives rise to the final generated plan.

3 Encoding New Problems In SIPE-2

SIPE-2 models the classical definition of the planning problem as previously explained in Section 2.1. Also previously noted, SIPE-2's form of input is a description of the initial state, a description of a set of actions, a problem descriptor, and a set of rules describing the domain. The task of encoding new problems to solve with SIPE-2 requires a working knowledge of the SIPE-2 language syntax (an extension of Backus-Naur Form (BNF) written by Donald Mitchell). Both a frame-based representation and first-order logic are used to describe the domain. The purpose of this section is to provide some basic guidelines and instructions for encoding domain knowledge as input to SIPE-2. The user (i.e., programmer) is referred to the SIPE-2 manual [Wilkins 93] and the book *Practical Planning* [Wilkins 88] for more detailed documentation on programming in the SIPE-2 language.

One of the first steps in defining the input for solving a new problem is to create a specification of the problem. One should identify the initial state, the goal state, and the nature of the actions that will advance the planner through intermediate states. For example, if the problem I want to solve is to evacuate all the people from a location X, my initial state is that everyone is at X, my goal state is that no-one is at X anymore, and in order to get from my initial state to my goal state, there will be movement from location X to some other location. Something to remember is that SIPE-2's output will be correct only to the extent that the user's representation correctly reflects the real environment. Additionally, the original specification of the problem may need modification or refinement in order to encode the problem input more efficiently.

The input information required by SIPE-2 is stored in a file, or multiple files, created by the user and must be loaded into the SIPE-2 system. This information consists of the following:

1. Definition of the sort hierarchy
2. Definition of the initial world predicates
3. Definition of the operators
4. Definition of the problem(s)

These topics will be discussed in more detail in the following subsections beginning with definition of the sort hierarchy.

3.1 Definition of the Sort Hierarchy

One input requirement of SIPE-2 is that it must be given information about the objects that exist in the domain. The sort hierarchy is the mechanism in SIPE-2 for representing these objects. As well as representing perpetual objects that exist in the domain, the sort hierarchy describes their invariant properties and their structure in terms of classes with inheritance properties.

The sort hierarchy is composed of class and object nodes. Classes define sets of objects with default properties common to all the individual objects belonging to that particular class. The links in the sort hierarchy should be interpreted as IS-A links meaning that each individual is an instance of the class or classes of which it is a member.

Properties of objects should be represented in the sort hierarchy if they are static and if they can be represented as unary or binary predicates. By representing domain objects along with most of their static properties as nodes linked in a sort hierarchy, SIPE-2 is able to incorporate the advantages of frame-based systems (e.g., inheritance, efficiency), while retaining the power of predicate calculus for representing properties that do vary. Properties of objects that might change as actions are performed (i.e., dynamic properties) and static properties that require more than two arguments should be represented as predicates in the restricted form of first-order predicate calculus provided in SIPE-2.

Class nodes can have the following slots: SUBCLASSES, INSTANCES, PARENT (or PARENT-CLASS), and PROPERTIES. Object nodes only have the slots: PARENT (or PARENT-CLASS), and PROPERTIES. An example of some class and object definitions is as follows:

> **CLASS:** Animals
> **Subclasses:** Lions, Tigers, Bears
> **END-CLASS**
>
> **CLASS:** Lions
> **Parent-Class:** Animals
> **Properties:**
> Habitat = Land;
> **Instances:** Lion1, Lion2
> **END CLASS**
>
> **OBJECT:** Lion1
> **Parent-Class:** Lions
> **Properties:**
> Color = Brown,
> Diet = Meat;
> **END OBJECT**

In this example, Lion1 is an instance of the class Lions with the properties that it has a color of brown and a diet of meat. Through inheritance from the parent-class Lions, Lion1 also has a habitat of land.

Once the objects in the world have been defined with the sort hierarchy, one must describe the initial state of the world. This is accomplished in the SIPE-2 language by defining initial world predicates as described next.

3.2 Definition of the Initial World Predicates

Initial world predicates are statements that describe the initial state of the world. Initial world predicates are divided into static and dynamic predicates depending on whether any of their truth values change over time as actions are performed. As stated in Section 3.1, properties of objects that might change as actions are performed should be represented as predicates, specifically dynamic predicates. Additionally, static properties of objects that cannot be represented as unary or binary predicates should be represented as static predicates.

Predicates can also be divided into closed and open predicates by whether or not SIPE-2 uses the closed-world assumption in reasoning about them. The closed-world assumption is the assumption that if an unnegated predicate is not given in the world model, then its negation is true. As an example, if one does not explicitly define a predicate such as (at Karen home), then the assumption is made that (not (at Karen home)) is true, meaning that Karen is not at home. The advantage of a closed predicate is that the user does not have to axiomatize every possible predicate instance that is not true in the domain. Without making the closed-world assumption (i.e., an open predicate is used), if the initial state is such that no one in New York State is at home, one would explicitly have to declare (not (at Karen home)), (not (at Bryan home)), (not (at Mary home)), etc., listing each resident of New York State explicitly.

3.3. Definition of the Operators

Operators refer to the system's representation of actions or abstractions of actions that may be performed in the world.

There are four types of operators: operator, state-rule, causal-rule and init-operator. An operator has a plot that describes how to expand a goal or process node to a lower level of detail. State rules, causal rules, and init-operators are collectively referred to as deductive rules, and have no plot. Deductive rules are used only to add new effects by doing a deduction. Init-operators are applied only once to deduce new statements about the initial world model. Causal-rules and state-rules both deduce new statements about the world and can be called recursively. The difference between causal and state rules is their order of applicability - causal rules are applied before state rules.

Operators can have the following slots: ID, TYPE, ARGUMENTS, RESOURCES, INSTANTIATE, PRECONDITION, CONDITION, PURPOSE, EFFECTS, TRIGGER, TIME-CONSTRAINTS, and PLOT. The ID slot is the name, or Lisp pointer, of the node. The TYPE slot stores the operator type, one of the four possibilities listed above. The ARGUMENTS slot is used to declare variables and constraints. The RESOURCES slot is a list of variables, generally a subset of the arguments slot, and implements the notion of reusable resource. Resources can also be declared for particular actions inside the plot, which is generally more useful. The INSTANTIATE slot is a list of variables, generally a subset of the arguments slot, which are instantiated by the system immediately after the operator is applied. The instantiation is done after the plot is inserted in the plan, but before deductions are made. The PRECONDITION slot is a list of predicates which, in regular operators, must be satisfied before the operator can be applied. In deductive rules, it is the antecedent of a deduction. Preconditions are goals the planner wants to be true but will not try to make true. If it is desirable to have the planner make them true, they must be put as goals in the plot of the operator. The CONDITION slot is a list of predicates, and is present only on deductive rules. In deductive rules, both conditions and preconditions are the antecedents of deductions but are matched under different circumstances. Conditions are always matched in the current state of the world, while preconditions are matched in the previous state. The PURPOSE slot, present only on regular operators, is a list of predicates that describes what the operator is trying to accomplish and is used to determine which goals the operator is able to expand. The EFFECTS slot is a list of predicates that represents the conclusions of a deduction or operator. If the preconditions and conditions of a deductive rule or operator hold, the effects of the rule or operator are added to the current state of the world. The TRIGGER slot, present only on deductive rules, is a predicate that triggers the application of that deductive rule when it is matched by a new effect. The TIME-CONSTRAINTS slot is a list of temporal constraints on the nodes in the plot of this operator. Each constraint consists of one of the thirteen Allen constraints [Allen 83] and the two nodes to which this constraint applies. The PLOT slot, present only on regular operators, provides the step-by-step instructions for performing the action represented by an operator. When expanding a plan to the next level of detail, SIPE-2 uses the plot as a template for generating nodes to insert in the plan. The plot consists of a network of goals or actions with associated orderings, conditions, and effects. A loop facility is provided for specifying parallel actions. Operators can also be expanded to a lower level with other operators that appear in its plot as process or choiceprocess nodes. The choiceprocess node is a way to specify a sequence of alternative operators to expand a certain operator.

The operators are tried in the order they are listed. Conditional plans can be specified in SIPE-2 with cond and condpattern nodes in the plot.

An example of an operator for picking up objects is shown below. The reader is referred to the SIPE-2 Manual [Wilkins 93] for a complete syntax for defining operators.

> **Operator:** pick-up-object
> **Arguments:** person1, object1, location1;
> **Purpose:** (pick-up person1 object1);
> **Precondition:** (at person1 location1),
> (at object1 location1);
> **Plot:**
> **Process**
> **Action:** pick-up
> **Arguments:** person1, object1;
> **Effects:** (pick-up person1 object1),
> (holding person1 object1);
> **End Plot End Operator**

There are some basic points to highlight concerning the definition and application of operators. One point is that there are two mechanisms for applying operators. Operators are invoked by their PURPOSE slot when the PURPOSE is matched to a goal that has been posted in the environment. An operator can also be invoked by its name when it is listed in the action slot of a choiceprocess node. A second point is that the EFFECTS slot is the list of new predicates that are added to the environment once the operator has been successfully applied. In order to use the Pick-Up-Object operator shown above, there must either be a goal posted in the problem definition, or by another operator, which matches the PURPOSE slot of the Pick-Up-Object operator, or second, another operator has been invoked which lists Pick-Up-Object in its plot. In the case of a posted goal, the goal must be something like (pick-up Karen Book) where Karen is a valid instantiation of Person1 and Book is a valid instantiation of Object1. When the goal (pick-up Karen Book) is matched with the PURPOSE slot (pick-up person1 object1), the Pick-Up-Object operator is selected to expand the goal. Assuming the preconditions of the operator are satisfied, the predicates listed in the EFFECTS slot are now written in the environment as (pick-up Karen Book) and (holding Karen Book).

3.4 Definition of the Problem(s)

The definition of the problem tells SIPE-2 what the system is attempting to solve. The problem definition specifies the goals that the planner must achieve, ranging from a single goal to a whole network of goals and actions. When SIPE-2 is given the problem to solve, the system searches through its operator definitions until it finds an operator whose PURPOSE slot matches the goal, or one of the goals, in the problem definition. SIPE-2 will select the first eligible operator that has been defined in the file. An example of a simple problem that SIPE-2 would be able to solve given the Pick-Up-Object operator described in Section 3.3 would look like the following:

> **Problem:** pick-up-an-object
> **Goal:** (Pick-up Karen Book)
> **End Problem**

Goals can be given sequentially or in parallel. As noted in Section 3.3, SIPE-2 provides a loop facility for specifying parallel actions in the plot of operators and problem definitions. Since SIPE-2 is a non-linear planner, part of its power lies in the fact that it can solve pieces of the whole problem in parallel.

An example of a problem with parallel goals that could be solved with the Pick-Up-Object operator is shown below:

Problem: pick-up-objects-in-parallel
  Parallel:
    Branch 1: Goal: (Pick-up Karen Book)
    Branch 2: Goal: (Pick-up Bryan Newspaper)
  End Parallel
End Problem

4 The Missionaries And Cannibals Puzzle

In order to learn the SIPE-2 system and to experiment with SIPE-2 in a meaningful way, we encoded the classical Artificial Intelligence (AI) Missionaries and Cannibals (MC) puzzle.

The MC problem statement is as follows:

There are three missionaries, three cannibals and a boat on the left bank of a river. To solve the puzzle you must transport all six persons to the right bank, using the boat. The boat only carries two people at a time, and at least one person must bring the boat back. The missionaries want to manage the trip across the river in such a way that the number of missionaries on either side of the river is never less than the number of cannibals who are on the same side.

The MC problem setup used for the exercise described in this report is as follows:

	Left bank	River		Right Bank
Initial state	MMMCCC	B		
Goal state			B	MMMCCC

Legal Moves				
	C	-->	<--	C
	CC	-->	<--	CC
	MC	-->	<--	MC
	MM	-->	<--	MM
	M	-->	<--	M

<u>Key to Notation</u>:

B	Boat
C	Cannibal
M	Missionary
-->	move to left
<--	move to right

Constraints: Cannibals must not outnumber missionaries on either bank.
Boat carries a maximum of two passengers.
At least one passenger must transport the boat back and forth.

5 Modeling the MC Puzzle in SIPE-2

This section discusses the MC puzzle as modeled in SIPE-2. Included in this discussion is our representation of the domain information required as input to the SIPE-2 system, e.g., the sort hierarchy, world predicates, operators, deductive rules, and problem definition for the MC puzzle. Further discussion of the encoding of the MC puzzle in SIPE-2 can be found in [Ludlow & Alguire 94].

The general structure of the MC puzzle as encoded in SIPE-2 closely follows the problem description given in Section 4. To increase the efficiency of the planner, we added the constraint that a previously visited state could not be re-visited. In other words, SIPE-2 could not use an operator that would return the system to a state already visited, thus preventing unnecessary loops. A complete listing of the SIPE-2 input for the MC puzzle is given as an appendix to this report.

5.1 The MC Puzzle Objects

The MC sort hierarchy is shown in Figure 3. The objects in the MC world consist of the following classes: People, River-banks, and Boats. The class People has the instances Cannibals and Missionaries. The class River-Banks has the instances Left-Bank and Right-Bank. The class Boats has the instance Two-People-Boat.

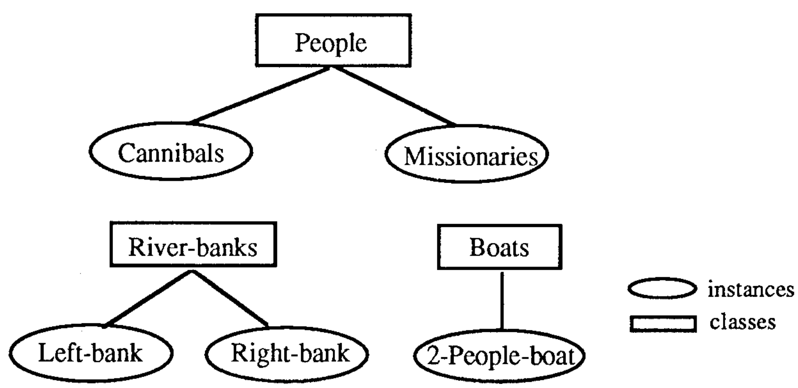

Figure 3. Sort Hierarchy for the MC Puzzle

5.2 The MC Puzzle Predicates

This section briefly describes the predicates defined for the MC puzzle. Figure 4 shows the initial state and the goal state for the MC puzzle.

The STATE predicate maintains information about the current state of the MC world. This predicate along with its negation prevents the same state from being visited more than once. This constraint was added to increase the efficiency of the planner by avoiding unnecessary looping. The STATE predicate has five arguments: the number of cannibals on the river bank where the boat is located, the number of missionaries on the river bank where the boat is located, the number of cannibals on the river bank opposite the boat, the number of missionaries on the river bank opposite the boat, and, lastly, the river bank where the boat is located. As an example of how the state predicate is used, the initial state in the MC world is given by the predicate (STATE 3 3 0 0 Left-bank).

When the MOVE-MC operator[3], which corresponds to move one missionary and one cannibal, is applied for the first move from the left bank to the right bank, the state predicate changes from (STATE 3 3 0 0 Left-bank) to (STATE 1 1 2 2 Right-bank).

Initial State	Goal State
(State 3 3 0 0 Left-bank)	
(Level Cannibals Left-bank 3)	
(Level Cannibals Right-bank 0)	
(Level Missionaries Left-bank 3)	(Move-Everybody)
(Level Missionaries Right-bank 0)	
(Opposite-bank Left-bank Right-bank)	
(Opposite-bank Right-bank Left-bank)	
(On Two-People-Boat Left-bank)	

Figure 4. Initial and Goal State for the MC Puzzle

The LEVEL predicate enables us to define the number of cannibals and missionaries on each bank. For example, the initial condition that there are 3 cannibals, 3 missionaries on the left and 0 cannibals, 0 missionaries on the right is represented with the following predicates:

(LEVEL Cannibals Left-Bank 3)
(LEVEL Cannibals Right-Bank 0)
(LEVEL Missionaries Left-Bank 3)
(LEVEL Missionaries Right-Bank 0)

The use of LEVEL predicates was necessary to force the instantiation of numerical variables in SIPE-2 so that numerical calculations and comparisons could be performed[4].

SIPE-2 has several built-in predicates related to Levels. The predicates PRODUCE and CONSUME are used to manipulate the levels of cannibals and missionaries by incrementing or decrementing by an amount dependent on which operator has been applied. The current value of a variable used in a LEVEL predicate can be checked by using the SIPE-2 predicates LEVEL<, LEVEL>, LEVEL>=, and LEVEL<=. This allowed us to guide when a particular operator(s) was to be used. For example, an operator to move a cannibal from right to left has preconditions to ensure that the level of cannibals on each bank is less than or equal to the level of missionaries on each bank before and after the operator is applied.

The OPPOSITE-BANK predicate allows us to identify banks that are opposite to one another. For example, the predicate (OPPOSITE-BANK Left-bank Right-bank) states that the Right-bank is opposite the Left-bank. This predicate is especially useful in the deductive rules defined for the MC puzzle since both sides of the river bank can be handled by a single rule.

[3]The operators for the MC puzzle are described in Section 5.3.
[4]Understanding the manipulation and instantiation of numerical variables in SIPE-2 was not a trivial task. We encountered some difficulties with numericals in the ported Sun version of SIPE-2 which arose from differences between Lucid Lisp numerical routines and Symbolics Lisp routines, the original SIPE-2 platform. The problems we encountered did not exist in the Symbolics version. The use of numericals in the MC puzzle is further discussed in Section 7.1 of this report.

The ON predicate is used to associate the boat with a river bank. For example, the initial condition that the boat is on the left bank is represented by (ON Two-People-Boat Left-Bank).

5.3 The MC Puzzle Operators

An overview of the MC puzzle operators is given in Figure 5. There are two levels of MC operators -- operators for controlling movement (SEND-TRIP-TO-RIGHT, SEND-TRIP-TO-LEFT) and operators for performing movement (MOVE-MC, MOVE-2-C, MOVE-2-M, MOVE-1-C, MOVE-1-M, END-MC). The following sections will briefly describe these operators in more detail. The complete definitions of the MC operators are given in the appendix.

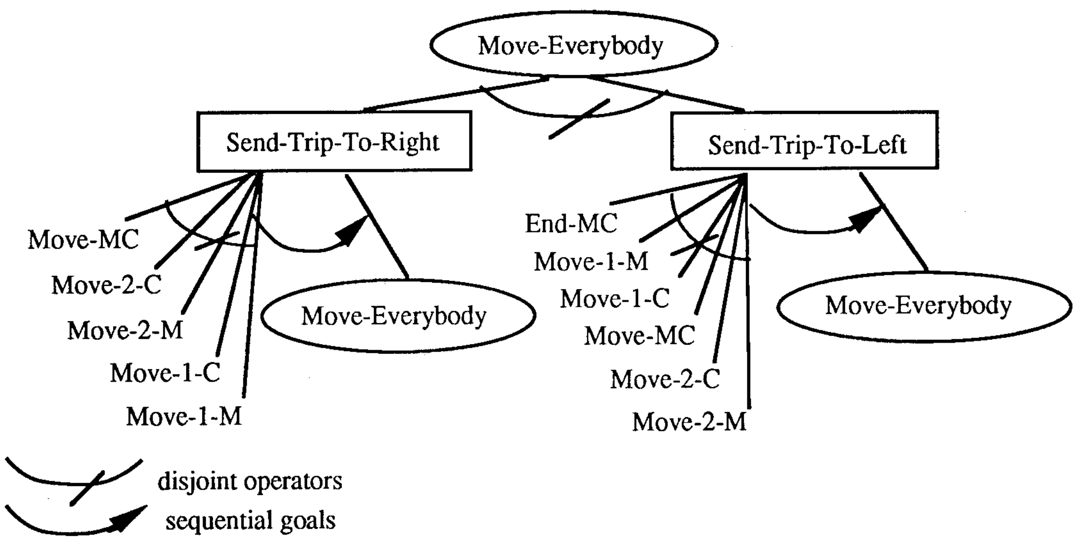

Figure 5. General Structure of the MC Puzzle Operators

5.3.1 Operators Controlling Movement

At the top level, there are two operators that control movement for the MC puzzle: SEND-TRIP-TO-RIGHT and SEND-TRIP-TO-LEFT. Both of these operators have as their purpose the predicate (Move-Everybody) and represent the flow of control for movement between the left and right river banks. The preconditions of the operators determine which one is used to expand the goal at a particular level of the plan. As an example, the SEND-TRIP-TO-RIGHT operator is shown below.

OPERATOR: SEND-TRIP-TO-RIGHT
ARGUMENTS: Continuous1, Numerical1 is previous value of Continuous1,
Continuous2, Numerical2 is previous value of Continuous2,
Continuous3, Numerical3 is previous value of Continuous3,
Continuous4, Numerical4 is previous value of Continuous4;
PURPOSE: (MOVE-EVERYBODY)
PRECONDITION: (ON Two-People-Boat Left-bank)
(LEVEL Cannibals Left-bank Continuous1)
(LEVEL Cannibals Right-bank Continuous2)

(LEVEL Missionaries Left-bank Continuous3)
(LEVEL Missionaries Right-bank Continuous4);

PLOT:
CHOICEPROCESS:
ACTION: MOVE-MC, MOVE-2-C, MOVE-2-M, MOVE-1-C, MOVE-1-M;
ARGUMENTS: Numerical1, Numerical3, Numerical2, Numerical4, Right-bank;
EFFECTS: (ON Two-People-Boat Right-bank);
GOAL: (MOVE-EVERYBODY)
END PLOT END OPERATOR

The operator SEND-TRIP-TO-RIGHT is used when the boat is on the left bank. The plot of the SEND-TRIP-TO-RIGHT operator is a choiceprocess node that lists the operators MOVE-MC, MOVE-2-C, MOVE-2-M, MOVE-1-C, and MOVE-1-M in its action slot. SIPE-2 will try to apply these operators in the order they have been given. In this way, we have communicated to SIPE-2 that our preference of operators for moving missionaries and cannibals from the left bank to the right are the ones that move two people first.

The four continuous variables listed in the arguments slot of the SEND-TRIP-TO-RIGHT operator are used because we want SIPE-2 to update the continuously changing number of missionaries and cannibals on the left and right banks in the LEVEL predicates. In general, this is the only situation in which continuous variables are to be used. In this particular operator, Continuous1 refers to the number of cannibals on the left bank, Continuous2 refers to the number of cannibals on the right bank, Continuous3 refers to the number of missionaries on the left bank, and Continuous4 refers to the number of missionaries on the right bank. The four numerical variables are used because we need to pass fixed numerical quantities to the operators listed in the choiceprocess node. These numerical variables are equal to the values of the four continuous variables just before the SEND-TRIP-TO-RIGHT operator is applied. Given the initial state of the MC puzzle as described in Section 5.2, the values of Numerical1, Numerical2, Numerical3, and Numerical4 the first time through are 3, 0, 3, 0, respectively. Since the MOVE-MC operator is the first one listed in the choiceprocess node, the SEND-TRIP-TO-RIGHT operator invokes MOVE-MC with five arguments representing the number of cannibals on the bank of origin, the number of missionaries on the bank of origin, the number of cannibals on the bank of destination, the number of missionaries on the bank of destination and the bank of destination. Since SEND-TRIP-TO-RIGHT initiates movement from left to right, the bank of origin in this case is the left bank and the bank of destination is the right bank. Thus, the first time through the MOVE-MC operator is invoked with the values 3, 3, 0, 0, Right-bank.

The operator SEND-TRIP-TO-LEFT is used when the boat is on the right bank. The definition of this operator is as follows:

OPERATOR: SEND-TRIP-TO-LEFT
ARGUMENTS: Continuous1, Numerical1 is previous value of Continuous1,
Continuous2, Numerical2 is previous value of Continuous2,
Continuous3, Numerical3 is previous value of Continuous3,
Continuous4, Numerical4 is previous value of Continuous4;
PURPOSE: (MOVE-EVERYBODY)
PRECONDITION: (ON Two-People-Boat Right-bank)
(LEVEL Cannibals Left-bank Continuous1)
(LEVEL Cannibals Right-bank Continuous2)
(LEVEL Missionaries Left-bank Continuous3)
(LEVEL Missionaries Right-bank Continuous4);
PLOT:

CHOICEPROCESS:
ACTION: END-MC, MOVE-1-M, MOVE-1-C, MOVE-MC, MOVE-2-C, MOVE-2-M;
ARGUMENTS: Numerical2, Numerical4, Numerical1, Numerical3, Left-bank;
EFFECTS: (ON Two-People-Boat Left-bank);
GOAL: (MOVE-EVERYBODY)
END PLOT END OPERATOR

The plot of the SEND-TRIP-TO-LEFT operator is a choiceprocess node whose action slot lists the operators END-MC, MOVE-1-M, MOVE-1-C, MOVE-MC, MOVE-2-C, MOVE-2-M. Notice that when moving from the right bank to the left, the preference is to use the operators that move only one person before using those that move two people. The arguments passed to the MOVE operators are the same as in SEND-TRIP-TO-RIGHT. However, since SEND-TRIP-TO-LEFT initiates movement from right to left, the order of the numerical variables passed to the MOVE operators is different to reflect the fact that the bank of origin is now the right bank and the bank of destination is now the left bank.

5.3.2 Operators Performing Movement

The operators that perform movement in the MC world imitate the legal moves of the Missionaries and Cannibals puzzle given in Section 4 of this report. These moves are summarized again in the following notation:

Legal Moves	C	-->	<--	C	<u>Key to Notation</u>:	
	CC	-->	<--	CC	C	Cannibal
	MC	-->	<--	MC	M	Missionary
	MM	-->	<--	MM	-->	move to left
	M	-->	<--	M	<--	move to right

As defined in the MC input file, the operators that move missionaries and cannibals are:

MOVE-MC	Move one missionary and one cannibal
MOVE-2-C	Move two cannibals
MOVE-1-C	Move one cannibal
MOVE-2-M	Move two missionaries
MOVE-1-M	Move one missionary
END-MC	Missionaries and Cannibals have all been moved

As an example, the MOVE-MC operator is listed below. The rest of the MOVE operators are structured in a similar manner. The first four arguments are numerical variables that represent the number of cannibals on the bank of origin, the number of missionaries on the bank of origin, the number of cannibals on the bank of destination, and the number of missionaries on the bank of destination. The fifth argument is the bank of destination. The values of these first five arguments are passed by either the SEND-TRIP-TO-RIGHT or SEND-TRIP-TO-LEFT operator as described in Section 5.3.1. Notice that the MOVE operators can handle movement in both directions. This is accomplished by changing the order in which the numerical variables are passed to the MOVE operators by SEND-TRIP-TO-RIGHT or SEND-TRIP-TO-LEFT.

OPERATOR: MOVE-MC
ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1,

```
                Numerical5 is (SIPE+ 1 Numerical3),
                Numerical6 is (SIPE+ 1 Numerical4),
                Numerical7 is (SIPE+ -1 Numerical1),
                Numerical8 is (SIPE+ -1 Numerical2), River-bank2;
PRECONDITION: (OPPOSITE-BANK River-bank1 River-bank2)
                (NOT (STATE Numerical5 Numerical6 Numerical7 Numerical8
                            River-bank1))
                (LEVEL>= Cannibals River-bank2 1)
                (LEVEL>= Missionaries River-bank2 1)
                (OR (LEVEL Missionaries River-bank2 1)
                    (LEVEL<= Numerical7 Numerical8))
                (LEVEL<= Numerical5 Numerical6);
PLOT:
  PROCESS
    ACTION: ROW-MC
    ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1;
    EFFECTS: (ON Two-People-Boat River-bank1)
              (PRODUCE Cannibals River-bank1 1)
              (PRODUCE Missionaries River-bank1 1)
              (STATE Numerical5 Numerical6 Numerical7 Numerical8
                     River-bank1);
END PLOT END OPERATOR
```

The MOVE-MC operator accepts the arguments passed to it and then calculates four numerical variables for reasoning about the new number of cannibals and missionaries on each river bank after accomplishing the move represented by the operator. In the MOVE-MC case, we are referring to the move of 1 cannibal and 1 missionary from the left to the right bank. The last argument is the river-bank of origin. The value of this argument is determined using the OPPOSITE-BANK predicate given the bank of destination. During the first pass through the MC puzzle, the values of Numerical1, Numerical2, Numerical3, and Numerical4 are instantiated as 3, 3, 0, 0, respectively, and the values of Numerical5, Numerical6, Numerical7, and Numerical8 are computed as 1, 1, 2, 2, respectively. SIPE+ is a built-in addition function provided by the system. Using the instantiated values of the variables, the preconditions of the MOVE-MC operator are checked for validity to see if the operator can be used or not. If the preconditions are satisfied, the effects of the MOVE-MC operator are evaluated.

When the PRODUCE predicates listed in the effects slot of the operator are posted to the environment, the levels of Cannibals and Missionaries on the right bank, as maintained by SIPE-2, are incremented by 1. As an example, the level of Cannibals on the right bank before using the PRODUCE predicate was 0 as given by the initial state predicate (LEVEL Cannibals Right-bank 0). After using the PRODUCE predicate, the level of Cannibals on the right bank becomes 1 which in SIPE-2 notation reads (LEVEL Cannibals Right-bank 1). Likewise for the level of Missionaries on the right bank. The PRODUCE events also trigger the two deductive rules described in Section 5.4 that use the CONSUME predicate to decrement the levels of Cannibals and Missionaries on the left bank by the same number. As an example, the initial predicate (LEVEL Cannibals Left-bank 3) becomes (LEVEL Cannibals Left-bank 2).

The four continuous variables used in the operators SEND-TRIP-TO-RIGHT and SEND-TRIP-TO-LEFT to represent the number of missionaries and cannibals as changing quantities are updated by the LEVEL predicate statements listed in the precondition slots of these operators. These updates reflect the changes of levels made by PRODUCE and CONSUME predicates in the MOVE operators. The updates to the continuous variables occur the next time the operators are applied.

When the number of missionaries and cannibals on the left bank is zero, SIPE-2 can apply the operator END-MC which signals that the problem is solved.

5.4 Domain Theory in the MC Puzzle

As previously mentioned, a powerful feature of SIPE-2 is the ability to encode a causal theory to represent and reason about the effects of actions in different world states. SIPE-2 uses deductive rules to allow effects of an event to be deduced without being mentioned explicitly in operators. Thus, the deduction of context-dependent effects is permitted. Deductive rules and their slots were briefly described in Section 3.3 of this report. Several causal rules that have been defined in the MC puzzle are shown below.

Causal-Rule: On-Bank-Not-On-Other-Bank
Arguments: Boat1, River-bank1, River-bank2;
Trigger: (ON Boat1 River-bank1);
Precondition: (ON Boat1 River-bank2) (OPPOSITE-BANK River-bank1 River-bank2);
Effects: (NOT (ON Boat1 River-bank2));
End Causal-Rule

Causal-Rule: Produce-and-Consume-Cannibals
Arguments: Cannibals, River-bank1, Numerical1, River-bank2;
Trigger: (PRODUCE Cannibals River-bank1 Numerical1);
Precondition: (OPPOSITE-BANK River-bank1 River-bank2)
Effects: (CONSUME Cannibals River-bank2 Numerical1);
End Causal-Rule

Causal-Rule: Produce-And-Consume-Missionaries
Arguments: Missionaries, River-bank1, Numerical1, River-bank2;
Trigger: (PRODUCE Missionaries River-bank1 Numerical1);
Precondition: (OPPOSITE-BANK River-bank1 River-bank2)
Effects: (CONSUME Missionaries River-bank2 Numerical1);
End Causal-Rule

The On-Bank-Not-On-Other-Bank rule has the effect that when the Two-People-Boat moves from one bank to the other bank, the boat is no longer on the original bank. This rule is triggered when an effect (ON Boat1 River-bank1) is posted by an operator during the planning process. The variables Boat1 and River-bank1 are instantiated by the trigger predicate. The OPPOSITE-BANK predicate determines the value of River-bank2 based on the value of River-bank1.

The Produce-And-Consume-Cannibals and the Produce-And-Consume-Missionaries rules help maintain how many cannibals and missionaries are on each bank by decreasing the number on one bank after there has been a move to the other bank. The Produce-And-Consume-Cannibals rule is triggered when an effect (PRODUCE Cannibals River-bank1 Numerical1) is posted. Likewise, the Produce-And-Consume-Missionaries rule is triggered when an effect (PRODUCE Missionaries River-bank1 Numerical1) is posted. River-bank1 is instantiated by the trigger predicate. The OPPOSITE-BANK predicate determines the value of River-bank2 based on the value of River-bank1. The use of the OPPOSITE-BANK predicate in these rules and the On-Bank-Not-On-Other-Bank rule above allows a single rule to handle both river banks. The number of cannibals and missionaries to subtract, either one or two, is represented by the variable Numerical1 and is also instantiated by the trigger predicate.

5.5 Problem Definition for the MC Puzzle

The last piece of information needed by SIPE-2 is the definition of the problem. The problem definition for this implementation of the MC puzzle is simply the goal (Move-Everybody) as shown below:

Problem: MC-Problem
Goal: (Move-Everybody)
End Problem

With the MC input as described in the previous sections, the SIPE-2 system has all the information needed to generate a plan. SIPE-2 can generate plans automatically or interactively via menus. The generation of a plan for solving the MC puzzle will be the topic of discussion in Section 6.

6 Generating SIPE-2 Plans for the MC Puzzle

This section briefly describes how SIPE-2 uses the MC puzzle input to generate plans for solving the MC problem. Details of using the SIPE-2 interface, as well as, explanations of system behavior can be found in [Wilkins 93]. When testing and running the SIPE-2 input file, it is helpful to run SIPE-2 interactively with various options in the Trace Profile Menu selected. The Trace Profile provides as detailed a trace of variable instantiations, goal phantomizations, and goal expansions as the user specifies, from high level notices to "gory details". The algorithm that SIPE-2 uses to produce new planning levels is described in Section 2.3 of this report.

When the goal (Move-Everybody) is posted as given in the MC-problem definition, SIPE-2 collects all the operators that can solve this goal, i.e., operators that have (Move-Everybody) as their purpose. The system finds the operators SEND-TRIP-TO-RIGHT and SEND-TRIP-TO-LEFT as described in Section 5.3.1. SIPE-2 begins with the first operator listed in the file, which in this case is SEND-TRIP-TO-RIGHT. If the preconditions are satisfied, the operator is used to expand the goal. If the preconditions are not satisfied, SIPE-2 goes to the next applicable operator in the file.

Given the initial state of the world as listed in Figure 4, the preconditions of SEND-TRIP-TO-RIGHT are satisfied. Thus, the SEND-TRIP-TO-RIGHT operator is selected and one level of planning is successfully completed.

As explained in Section 5.3.1, the first item in the plot of the SEND-TRIP-TO-RIGHT operator is a choiceprocess node listing the operators MOVE-MC, MOVE-2-C, MOVE-2-M, MOVE-1-C, and MOVE-1-M. SIPE-2 expands the choiceprocess node by invoking these operators, with the specified arguments, in the order they have been listed until finding one whose preconditions are all satisfied. Since the MOVE-MC operator is listed first, SIPE-2 checks the preconditions to see if this operator can be used. Since the preconditions of this operator are satisfied, SIPE-2 selects the MOVE-MC operator. SIPE-2 successfully applies MOVE-MC and the instantiated effects of the operator are now added to the environment. For sake of clarity, these effects are listed as follows:

(ON Two-People-Boat Right-bank)
(PRODUCE Cannibals Right-bank 1)
(PRODUCE Missionaries Right-bank 1)
(STATE 1 1 2 2 Right-bank)

Section 5.4 described three deductive rules that were defined for the MC puzzle. When each operator effect is added to the environment, SIPE-2 checks for any deductive rules that can be triggered. When the first effect is added to the environment, the deductive rule On-Bank-Not-On-Other-Bank is triggered which negates the predicate (ON Two-People-Boat Left-bank) since a boat can only be on one bank at a time. When the second effect is added, the deductive rule Produce-And-Consume-Cannibals is triggered. The third effect triggers the Produce-And-Consume-Missionaries rule. Consequently, one cannibal and one missionary are automatically subtracted from the left river bank. Without the deductive rules, each of these "automatic" effects would have to be explicitly listed in every operator that causes them.

SIPE-2 control returns to the SEND-TRIP-TO-RIGHT operator, which posts the goal (Move-Everybody) a second time. The process of achieving the (Move-Everybody) goal begins again, only now the state of the world has been updated to reflect the fact that a missionary, a cannibal, and the boat have moved from the left bank to the right bank.

SIPE-2 tries to apply the SEND-TRIP-TO-RIGHT operator again. However, this time the preconditions fail because the boat is no longer on the left river-bank. SIPE-2 continues with the next operator which is SEND-TRIP-TO-LEFT. The preconditions of SEND-TRIP-TO-LEFT are satisfied, thus, SIPE-2 finds a valid plan at level 2.

The operators listed in the choiceprocess node of SEND-TRIP-TO-LEFT are END-MC, MOVE-1-M, MOVE-1-C, MOVE-MC, MOVE-2-C, MOVE-2-M. The preconditions for the END-MC operator are not satisfied, so SIPE-2 tries the MOVE-1-M operator next. The preconditions for MOVE-1-M are satisfied so SIPE-2 selects this operator. After MOVE-1-M is successfully applied, the effects of the operator, along with deduced effects, are added to the environment. SIPE-2 control returns to the SEND-TRIP-TO-LEFT operator which posts the goal (Move-Everybody) and the process repeats. SIPE-2 continues until all the missionaries and cannibals have moved to the right bank. When the level of missionaries and cannibals on the left is zero, the preconditions of the END-MC operator are satisfied and the MC problem is solved.

SIPE-2 solves this implementation of the MC problem in 13 planning levels. When the option "Backtrack To Find Alternatives" is selected from the Plan Automatic menu, SIPE-2 generates four plan solutions as listed below.

```
;;; Solution 1

ROW-MC   3 3 0 0  RIGHT-BANK
ROW-1-M  1 1 2 2  LEFT-BANK
ROW-2-C  2 3 1 0  RIGHT-BANK
ROW-1-C  3 0 0 3  LEFT-BANK
ROW-2-M  1 3 2 0  RIGHT-BANK
ROW-MC   2 2 1 1  LEFT-BANK
ROW-2-M  2 2 1 1  RIGHT-BANK
ROW-1-C  1 3 2 0  LEFT-BANK
ROW-2-C  3 0 0 3  RIGHT-BANK
ROW-1-M  2 3 1 0  LEFT-BANK
ROW-MC   1 1 2 2  RIGHT-BANK
EVERYBODY-MADE-IT

;;; Solution 2

ROW-MC   3 3 0 0  RIGHT-BANK
ROW-1-M  1 1 2 2  LEFT-BANK
ROW-2-C  2 3 1 0  RIGHT-BANK
ROW-1-C  3 0 0 3  LEFT-BANK
ROW-2-M  1 3 2 0  RIGHT-BANK
ROW-MC   2 2 1 1  LEFT-BANK
ROW-2-M  2 2 1 1  RIGHT-BANK
ROW-1-C  1 3 2 0  LEFT-BANK
ROW-2-C  3 0 0 3  RIGHT-BANK
ROW-1-C  2 3 1 0  LEFT-BANK
ROW-2-C  2 0 1 3  RIGHT-BANK
EVERYBODY-MADE-IT

;;; Solution 3

ROW-2-C  3 3 0 0  RIGHT-BANK
ROW-1-C  2 0 1 3  LEFT-BANK
ROW-2-C  2 3 1 0  RIGHT-BANK
ROW-1-C  3 0 0 3  LEFT-BANK
ROW-2-M  1 3 2 0  RIGHT-BANK
ROW-MC   2 2 1 1  LEFT-BANK
ROW-2-M  2 2 1 1  RIGHT-BANK
ROW-1-C  1 3 2 0  LEFT-BANK
ROW-2-C  3 0 0 3  RIGHT-BANK
ROW-1-M  2 3 1 0  LEFT-BANK
ROW-MC   1 1 2 2  RIGHT-BANK
EVERYBODY-MADE-IT

;;; Solution 4

ROW-2-C  3 3 0 0  RIGHT-BANK
ROW-1-C  2 0 1 3  LEFT-BANK
ROW-2-C  2 3 1 0  RIGHT-BANK
ROW-1-C  3 0 0 3  LEFT-BANK
ROW-2-M  1 3 2 0  RIGHT-BANK
ROW-MC   2 2 1 1  LEFT-BANK
ROW-2-M  2 2 1 1  RIGHT-BANK
ROW-1-C  1 3 2 0  LEFT-BANK
ROW-2-C  3 0 0 3  RIGHT-BANK
ROW-1-C  2 3 1 0  LEFT-BANK
ROW-2-C  2 0 1 3  RIGHT-BANK
EVERYBODY-MADE-IT
```

The fields of the plan solutions represent the following:

1) the action specified in the ACTION slot of the operator selected by SIPE-2
2) the number of cannibals on the bank of origin before the move
3) the number of missionaries on the bank of origin before the move

4) the number of cannibals on the bank of destination before the move
5) the number of missionaries on the bank of destination before the move
6) the bank of destination

Looking at the first line of Solution 1, we see that the ACTION was ROW-MC which corresponds to the selection of the MOVE-MC operator, the number of cannibals on the left bank was 3, the number of missionaries on the left bank was 3, the number of cannibals on the right bank was 0, the number of missionaries on the right bank was 0, and the bank of destination was the right bank. The second line of Solution 1 shows that the MOVE-1-M operator was chosen next, the number of cannibals on the right was 1, the number of missionaries on the right was 1, the number of cannibals on the left was 2, the number of missionaries on the left was 2, and the bank of destination was the left bank. The third operator selected was MOVE-2-C with a destination of the right bank, and so on.

7 Lessons Learned

The implementation of the MC puzzle as described in this report is the product of several iterations. We encountered some problems implementing the MC puzzle in SIPE-2 related to the use of numerical variables, the encoding of operators with multiple effects, and goal phantomization through the instantiation of variables. These issues will be discussed in light of our experience in an effort to supplement the SIPE-2 documentation currently available on the topics.

7.1 The Use of Numerical Variables

Numerical variables are distinguished into two classes, Numerical and Continuous. The numerical class is used for variables that will eventually be instantiated to one particular number. Variables in the continuous class will have values that vary with time as actions are performed and must be computed by the system. A general rule is to always use the class Numerical except when you want to force SIPE-2 to compute the value of a continuously changing quantity. SIPE-2 provides the special predicates PRODUCE, CONSUME, LEVEL, LEVEL<, LEVEL>, LEVEL<=, and LEVEL>= for numerical reasoning.

There are five numerical constraints that may be posted on variables of the Class Numerical (i.e., not on continuous variables nor any nonnumerical variables). These constraints are current-value, previous-value, range, function, and summary-range. Other nonnumerical constraints may also be posted on numerical variables. The Current-value constrains a numerical variable to be the current value of a continuous variable at a particular point in the plan and is taken just after the current node. Similarly, the previous-value constrains a numerical variable to be the previous value of a continuous variable at a particular point in the plan and is taken just before the current node. The Range constrains a variable to lie within a given range. Function constrains a variable to be the value of a certain function applied to any number of arguments. If some of these arguments are not instantiated, SIPE-2 computes a range from the function constraint by calling the function on all the possible instantiations. When the function constraints are used, SIPE-2 expects the name of a Lisp function or the name of an attribute in the sort hierarchy. The arguments to the Lisp function can be any SIPE-2 argument, and it should return either T or a SIPE-2 numerical object. Computing the constraints described above can be expensive. Thus, it is sometimes necessary to make tradeoffs between storage and recomputation. The SIPE-2 system handles this problem by posting the summary-range constraint on numerical variables which stores the results of computing a noncontinuous variable's constraints. The summary-range constraint can be posted only by the system, not by users.

Compared to other AI planners, SIPE-2 has a fairly extensive numerical reasoning capability. SIPE-2 has mechanisms for reasoning about numerical quantities, both continuous and discrete. The user invokes these mechanisms by declaring variables to be in the numerical class, and using constraints and certain system-defined predicates on them as described above. Numericals have been used in domains such as beer production planning and military crisis action planning to represent start times, end times, durations, production levels, mobilities, runway lengths, etc.

Nevertheless, understanding and using numericals in SIPE-2 is not a trivial task. We spent some time trying to understand the use, manipulation, and instantiation of numerical variables and especially how to properly and efficiently implement even the simple numerical constraints in our representation of the MC puzzle. The differentiation and usage of numerical vs. continuous variables was not clear based on available documentation and examples. Further experimentation and discussion with the system developer was required for clarification on this topic. In addition, an issue that requires further investigation is how numerical reasoning capabilities in SIPE-2 scale up to handle numerical constraints inherent in applications of real world magnitude, e.g., the transportation of military forces, where the number of people to transport is on the order of thousands rather than single digits as in the MC puzzle.

7.2 The Encoding of Operators with Multiple Effects

As an initial attempt to formalize the MC puzzle in SIPE-2, we defined the class hierarchy for missionaries and cannibals differently than the current representation described in this report. The Class Persons had as subclasses the classes Cannibals and Missionaries. The Class Cannibals had three instances "C1", "C2", and "C3" to represent the three cannibals in the puzzle. Similarly, the Class Missionaries had three instances "M1", "M2", and "M3" to represent the three missionaries in the puzzle. The initial state of the puzzle was represented by a set of seven predicates of the form (ON x Left-bank), where x corresponded to the cannibals, missionaries, and the boat. As an example, the predicate (ON C1 Left-bank) declares that the first cannibal is on the left bank. The goal state was represented by a set of six predicates of the form (ON x Right-bank), where x corresponded to the cannibals and the missionaries. In this formalization, there were ten main operators corresponding to legal moves. For example, the operator Move-2C-L-R corresponded to the legal move of two cannibals riding in the boat from the left bank to the right bank of the river. The other operators were Move-MC-L-R, Move-MC-R-L, Move-1M-L-R, Move-1M-R-L, Move-2M-L-R, Move-2M-R-L, Move-1C-L-R, Move-1C-R-L, and Move-2C-R-L.

The initial problem definition was structured as follows:

```
Problem: MC-Problem
  Parallel:
    Branch 1: Goal: (On C1 Right-bank)
    Branch 2: Goal: (On C2 Right-bank)
    Branch 3: Goal: (On C3 Right-bank)
    Branch 4: Goal: (On M1 Right-bank)
    Branch 5: Goal: (On M2 Right-bank)
    Branch 6: Goal: (On M3 Right-bank)
  End Parallel
End Problem
```

The difficulty we encountered involved the encoding of the operators that moved two people, as in the operator Move-2C-L-R shown below. The purpose of the operator was to move two different cannibals from the left to the right bank, cannibal1 and cannibal2 which is not cannibal1.

```
OPERATOR: MOVE-2C-L-R
ARGUMENTS: Cannibal1, Right-bank, Cannibal2 is not Cannibal1, Numerical1,
           Numerical2, Numerical3, Numerical4, Numerical5, Numerical6,
           Two-people-boat, Left-bank, Missionaries-On-Right-Bank;
PRECONDITION: (ON Cannibal1 Left-bank),
              (ON Cannibal2 Left-bank),
              (OR (LEVEL Missionaries-On-Right-Bank 0)
                  (LEVEL>= Missionaries-On-Right-Bank Numerical6)),
              (NOT (STATE Numerical5 Numerical6 Numerical3 Numerical4));
PLOT:
  PROCESS
```

ACTION: MOVE-2C-L-R-PRIM
ARGUMENTS: Cannibal1, Right-bank, Cannibal2, Left-bank, Two-people-boat, Numerical5, Numerical6, Numerical3, Numerical4, Cannibals-on-Left-Bank, Cannibals-on-Right-Bank;
EFFECTS: (ON Cannibal1 Right-bank),
(NOT (ON Cannibal1 Right-bank)),
(ON Cannibal2 Right-bank),
(NOT (ON Cannibal2 Right-bank)),
(ON Two-people-boat Right-bank),
(NOT (ON Two-people-boat Right-bank)),
(STATE Numerical5, Numerical6, Numerical3, Numerical4),
(CONSUME Cannibals-on-Left-Bank 2),
(PRODUCE Cannibals-on-Right-Bank 2);
END PLOT END OPERATOR

In order to check the correctness of our operator, we isolated the task of moving two cannibals. The simplified problem definition consisted of two sequential goals as follows:

Problem: MC-Problem
Goal: (On C1 Right-bank)
Goal: (On C2 Right-bank)
End Problem

When SIPE-2 applied the MOVE-2C-L-R operator to expand the first problem goal, (ON C1 Right-bank), the relevant operator effects that were posted were (ON C1 Right-bank), where Cannibal1 was instantiated to the instance C1, and the secondary effect (ON Cannibal2 Right-bank) where Cannibal2 was an uninstantiated variable constrained to be a cannibal which was not C1. One would have expected the posted secondary effect (ON Cannibal2 Right-bank) to satisfy the second goal of our simplified MC problem (ON C2 Right-bank). However, SIPE-2 would not match the uninstantiated secondary effect posted by the MOVE-2C-L-R operator to the goal (ON C2 Right-bank). Thus, SIPE-2 would fail to solve the problem. This seemed to be related to the fact that the secondary effect variable was not directly involved in the goal that was originally posted. A possible explanation has to do with the way SIPE-2 recognizes goals that are already achieved (i.e., the goal predicate is true) at the point in the plan where they occur. This is referred to as goal phantomization.

The problems we were experiencing led us to adapt an alternative representation of the problem, the one described in this report, which was also more efficient than the initial formulation. We have since experimented more with variable instantiation and, related to this, the phantomization of goals through variable instantiation which will be discussed next.

7.3 The Phantomization of Goals Through Variable Instantiation

As mentioned earlier in this report, SIPE-2 uses variables and the ability to post constraints on variables in the planning process. While this is a desirable feature of generative planners, it also introduces some complexity because the planner must determine when a predicate containing a variable is true. The use of variables also introduces complexity in what is referred to as goal phantomization, i.e., the process of achieving a goal by having it already true at the point in the plan where it occurs. This section will discuss this issue in relation to some of the problems we experienced regarding our initial formalization of the MC puzzle in SIPE-2.

In general, SIPE-2 adheres to its least-commitment philosophy by refusing to instantiate variables or to add ordering constraints unless forced to do so. Regarding the use of instantiation to accomplish goals, the system provides several options. These options are as follows: to instantiate whenever possible, to never post constraints, or to instantiate only when there is one possible binding to solve the goal. The first choice does not actually instantiate variables unless there is only one possible instantiation to accomplish the goal. This is the choice most often used in the domains implemented in SIPE-2. The second choice avoids commitment by never posting constraints of any kind. The third choice allows constraints which instantiate a variable to a given object (the Instan constraint) or to the same object to which another variable has been instantiated (the Same constraint). However, it does not allow the Pred constraint which must instantiate a variable so that a given predicate containing that variable is true.

Options for using helpful interactions between parallel branches are more restricted since adding ordering constraints is not irrevocable and instantiations may have to be made. If two branches of a plan are in parallel, an interaction is defined to occur when an expected effect (i.e., any formula the planner expects to be true within a subplan) in one branch possibly codesignates[5] with an expected effect in another branch. Helpful interactions are effects that agree in sign -- meaning they are both negated, both unnegated, or both unknown. The three options for using helpful interactions provided by SIPE-2 are as follows: to never add ordering constraints for the purpose of phantomization, to add ordering constraints when no other constraints are required to accomplish the phantomization, or to add ordering constraints when there is a parallel branch with an effect that has only one possible instantiation for accomplishing the phantomization. The second option means that the helpful effect in a parallel branch is necessarily codesignated with the goal being phantomized and that the goal to be phantomized is guaranteed to be true at the end of the parallel branch. The third choice also requires that the goal to be phantomized is guaranteed to be true at the end of the parallel branch, but allows the effect to be possibly codesignated as long as there is only one possible instantiation that will make it codesignate.

If the effects between parallel branches do not agree in sign, they may be harmful interactions. SIPE-2 has several techniques for dealing with this problem. The most important one is SIPE-2's ability to use resource reasoning to handle many problems that other classical planners would have to resolve through harmful parallel interactions. Another important technique for dealing with nonlinear actions is distinguishing between main effects and side effects of an action. SIPE-2 recognizes and resolves only those interactions that deal with the main effects of nodes. Since SIPE-2 provides flexibility in specifying main and side effects, problematic predicates can be changed to main effects. This greatly reduces the computational burden because the system is not required to resolve conflicts that do not matter. By default, all effects that are deduced are considered to be side effects. Effects that are provided by the goals of the problem or are introduced directly by operator applications are main effects. SIPE-2 also simplifies the problem by not shuffling actions between two parallel branches. It only orders the actions by putting one branch before or after the others. While this prevents some solutions from being found (as in the MC operator example described above), it is a technique for retaining efficiency while not being overly restrictive.

[5]The term *possibly codesignates* means that two variables/objects unify, i.e., it is consistent to assume they have the same instantiation. Alternatively, the term *necessarily codesignates* means that two variables/objects are the same objects, are instantiated to the same objects or are constrained to be instantiated to the same objects. The term codesignates used without a modifier means that the objects in question *necessarily* codesignate. Possible codesignation is referred to explicitly.

When it was determined that the problems encountered encoding operators with multiple effects was related to goal phantomization in SIPE-2, some tests were performed regarding the way SIPE-2 handles this issue. [Ludlow 94] describes a simplified problem analogous to the situation detected when we were trying to implement the operators moving two people and presents a summary of a goal phantomization analysis performed in SIPE-2. The situation tested involved a single operator with two main effects -- the first main effect matching the purpose of the operator while the second main effect did not match the purpose of the operator. Additionally, the second main effect was either a predicate of arity zero (i.e., a constant), or a predicate of arity one (i.e., a predicate involving one variable). If the second main effect involved a variable, that variable was not involved in the purpose of the operator. All of the problems consisted of two goals -- the first goal matching the purpose of the operator and the second goal matching the secondary effect of the main operator. Since the variables involved in the secondary effect of the operator were different than the variables involved in the purpose of the operator, there was no guarantee that SIPE-2 would instantiate them so as to phantomize the second goal. Thirteen variations of the above situation, as well as, two types of problems were considered -- one with sequential goals and one with parallel goals. The purpose of the experiment conducted by Ludlow was to determine under what conditions the second goal would be (or not be) phantomized as a result of the application of the operator to achieve the first goal. To briefly summarize, 4 problems could be solved only sequentially, 4 problems could be solved both sequentially and in parallel, and 5 problems could not be solved either sequentially or in parallel. Thus, SIPE-2 fails to solve 9 out of 13 of the parallel problems, and 5 out of 13 of the sequential problems. The results of this investigation are more fully documented in the reference previously given.

8 Conclusion

This report describes work related to some experiments performed at Rome Laboratory with generative planners. The main objective of the project was to provide an understanding of generative planners and familiarization with two different generative planners, O-Plan2 and SIPE-2. A secondary objective of the project, mainly a subproduct of the experimentation with both planners, was the comparison between O-Plan2 and SIPE-2. In particular, this report highlights our experimentation with the SIPE-2 planning system in terms of encoding the Missionaries and Cannibals puzzle, a simple classical problem in Artificial Intelligence. This report also provides guidance for defining and solving new problems with SIPE-2 in an effort to capture our experience with the encoding process.

The Missionaries and Cannibals puzzle was selected for our experimentation with generative planners because it is a simple AI classical planning problem and it deals with numerical calculations, an aspect we felt was relevant to application interests. The encoding of the Missionaries and Cannibals puzzle in SIPE-2 was instrumental in understanding the implementation of fundamental generative planning capabilities, as well as, the overall planning process embodied in the SIPE-2 system. In addition, the MC puzzle provided us with a frame of reference with which to interact with system developers to pinpoint strengths and limitations of the system. While SIPE-2 is a fairly mature system, we discovered that it is also still a research product with room for improvement and further testing. As a result of our effort encoding the MC puzzle, several errors were detected and corrected with patches that were added to the system. Several discussions regarding the state of current AI planners were spawned in an attempt to speculate whether generative planning technology was ready to handle the complexity and scale of real world problems.

We encountered some problems encoding the MC puzzle related to the use of numerical variables, the encoding of operators with multiple effects, and goal phantomization through the instantiation of variables. These issues are discussed in light of the MC encoding exercise. Among AI planners, SIPE-2 is one of the pioneers in manipulating numerical quantities. SIPE-2 has mechanisms for reasoning about numerical quantities, both continuous and discrete. The user invokes these mechanisms by declaring variables to be in the numerical class, and using constraints and certain system-defined predicates on them. However, it took some time understanding the use and manipulation of numerical variables to properly implement even the simple numerical constraints in our representation of the MC puzzle.

Another area of investigation was the way SIPE-2 handles the phantomization of goals through variable instantiation. Before fully understanding the heuristics embodied in SIPE-2, we experienced some difficulties encoding operators with multiple effects. This led us to look more closely at variable instantiation and the goal phantomization process. Several example files were created to exercise our understanding of these concepts. A summary of the results regarding this experimentation can be found in [Ludlow 94]. Ludlow presents some situations in which SIPE-2 fails to phantomize goals due to its heuristics. A topic of further investigation would be to determine how well the heuristics employed in SIPE-2 perform in general.

The final MC puzzle input solution documented in this report is the result of several iterations and refinements. The task of learning to encode operators in SIPE-2 is fairly straightforward as long as one takes the time to carefully read and study available documentation and existing examples. However, encoding operators in SIPE-2 to produce an efficient plan solution requires a fair amount of expertise with the SIPE-2 language formalism.

As a result of this exercise, we examined several fundamental aspects of generative planning. These aspects include the encoding and expansion of operator schemata, the instantiation and manipulation of numerical variables, the use and encoding of operators with multiple effects, the implementation of a causal theory through deductive reasoning, goal phantomization, and search. Overall, this project has been a valuable learning experience concerning the details of the SIPE-2 planning system and, more generally, in the area of generative planning as a whole.

9 Acknowledgments

The author would like to thank Carla Ludlow whose experience and determination made this project very rewarding, David Wilkins for his helpful comments and suggestions for using SIPE-2, and Nort Fowler for his support and interest in this project.

References

[Allen 83] J. F. Allen, "Maintaining Knowledge About Temporal Intervals," *Communications of the Association for Computing Machinery*, Vol. 26 No. 11, 1983.

[Arthur *et al.* 93] Arthur, R., Deitsch, A., and Stillman, J., "Tachyon: A Constraint-based Temporal Reasoning Model and its Implementation," SIGART Bulletin, 4:3, July 1993.

[Currie & Tate 91] Currie, K.W., and Tate, Austin, "O-Plan: The Open Planning Architecture", *Artificial Intelligence*, 51(1), Autumn 1991.

[Ludlow 94] Ludlow, Carla O., "O-Plan2 vs. SIPE-2 - A General Comparison," RL-TR-94-96, Rome Laboratory, Griffiss AFB, New York, 1994.

[Ludlow & Alguire 94] Ludlow, Carla O., and Alguire, Karen M., "Looking at O-Plan2 and SIPE-2 Through the Missionaries and Cannibals," *Proceedings of the 1994 ARPA/RL Knowledge-Based Planning and Scheduling Initiative Workshop*, 1994.

[O-Plan2 93] *O-Plan2: The Open Planning Architecture, User Guide Version 2.1*, AIAI, University of Edinburgh, 1993.

[Wilkins 88] Wilkins, David E., *Practical Planning: Extending the Classical AI Planning Paradigm,,* Morgan Kaufmann Publishers, Inc., San Mateo, CA., 1988.

[Wilkins 93] Wilkins, David E., *Using the SIPE-2 Planning System: A Manual for SIPE-2, Version 4.3*, SRI International, 1993.

[Wilkins *et al.* 94] Wilkins, David, Myers, Karen, Wesley, Leonard, and Lowrance, John, *Planning in Dynamic and Uncertain Environments*, RL-TR-94-39, Rome Laboratory, Griffiss AFB, New York, 1994.

Appendix

Implementation of the Missionaries and Cannibals Puzzle in SIPE-2[6]

```
;AUTHOR: Carla O. Ludlow and Karen M. Alguire and David E. Wilkins
;DESCRIPTION: the missionaries and cannibals problem

;;===================== Sort Hierarchy ========================

(DEFINE.DOMAIN)

CLASS: River-Banks
INSTANCES: Left-bank, Right-bank
END CLASS

CLASS: People
INSTANCES: Cannibals, Missionaries
END CLASS

CLASS: Boats
INSTANCES: Two-People-Boat
END CLASS

STOP

;;================== Initial World Predicates ========================

(DEFINE.DOMAIN)

static-predicates: (OPPOSITE-BANK)

PREDICATES:

(OPPOSITE-BANK Left-bank Right-bank)
(OPPOSITE-BANK Right-bank Left-bank)

(ON Two-People-Boat Left-bank)

(STATE  3 3 0 0 Left-bank)

(LEVEL Cannibals Left-bank 3)
(LEVEL Cannibals Right-bank 0)
(LEVEL Missionaries Left-bank 3)
(LEVEL Missionaries Right-bank 0)

END PREDICATES

STOP

;;=======================OPERATORS========================

(DEFINE.DOMAIN)

OPERATOR: SEND-TRIP-TO-RIGHT
```

[6]The implementation of the MC puzzle as listed in this appendix is a revised version of an original solution encoded by Carla Ludlow and Karen Alguire. David Wilkins, the developer of SIPE-2, made several suggestions to achieve a more efficient solution, the version that is listed here.

```
ARGUMENTS: Continuous1, Numerical1 is previous value of Continuous1,
           Continuous2, Numerical2 is previous value of Continuous2,
           Continuous3, Numerical3 is previous value of Continuous3,
           Continuous4, Numerical4 is previous value of Continuous4;
PURPOSE: (MOVE-EVERYBODY)
PRECONDITION: (ON Two-People-Boat Left-bank)
              (LEVEL Cannibals Left-bank Continuous1)
              (LEVEL Cannibals Right-bank Continuous2)
              (LEVEL Missionaries Left-bank Continuous3)
              (LEVEL Missionaries Right-bank Continuous4);
PLOT:
  CHOICEPROCESS:
   ACTION: MOVE-MC, MOVE-2-C, MOVE-2-M, MOVE-1-C, MOVE-1-M;
   ARGUMENTS: Numerical1, Numerical3, Numerical2, Numerical4, Right-bank;
   EFFECTS: (ON Two-People-Boat Right-bank);
   GOAL: (MOVE-EVERYBODY)
END PLOT END OPERATOR


OPERATOR: SEND-TRIP-TO-LEFT
ARGUMENTS: Continuous1, Numerical1 is previous value of Continuous1,
           Continuous2, Numerical2 is previous value of Continuous2,
           Continuous3, Numerical3 is previous value of Continuous3,
           Continuous4, Numerical4 is previous value of Continuous4;
PURPOSE: (MOVE-EVERYBODY);
PRECONDITION: (ON Two-People-Boat Right-bank)
              (LEVEL Cannibals Left-bank Continuous1)
              (LEVEL Cannibals Right-bank Continuous2)
              (LEVEL Missionaries Left-bank Continuous3)
              (LEVEL Missionaries Right-bank Continuous4);
PLOT:
  CHOICEPROCESS:
    ACTION: END-MC, MOVE-1-M, MOVE-1-C, MOVE-MC, MOVE-2-C, MOVE-2-M;
    ARGUMENTS: Numerical2, Numerical4, Numerical1, Numerical3, Left-bank;
    EFFECTS: (ON Two-People-Boat Left-bank);
    GOAL: (MOVE-EVERYBODY)
END PLOT END OPERATOR


OPERATOR: END-MC
ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1;
PRECONDITION: (LEVEL Missionaries Left-bank 0)
              (LEVEL Cannibals Left-bank 0);
PLOT:
  PROCESS ACTION: EVERYBODY-MADE-IT
      EFFECTS: (MOVE-EVERYBODY);
END PLOT END OPERATOR

;;=========================================================
;;; The Send-Trip-to-Right and Send-Trip-to-Left operators instigate trips by (1) listing
;;  preferred ordering of operators (prefer 2 people when moving to goal, 1 person on
;;  return trips), and (2) computing numerical values of state for use in move oprs.


;;================== Move Operators ==========================

OPERATOR: MOVE-MC
ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1,
```

```
                 Numerical5 is (SIPE+ 1 Numerical3),
                 Numerical6 is  (SIPE+ 1 Numerical4),
                 Numerical7 is (SIPE+ -1 Numerical1),
                 Numerical8 is  (SIPE+ -1 Numerical2),  River-bank2;
PRECONDITION: (OPPOSITE-BANK River-bank1 River-bank2)
                 (NOT (STATE Numerical5 Numerical6 Numerical7 Numerical8
                              River-bank1))
                 (LEVEL>= Cannibals River-bank2 1)
                 (LEVEL>= Missionaries River-bank2 1)
                 (OR (LEVEL Missionaries River-bank2 1)
                     (LEVEL<= Numerical7 Numerical8))
                 (LEVEL<= Numerical5 Numerical6);
PLOT:
  PROCESS
    ACTION: ROW-MC
    ARGUMENTS:  Numerical1, Numerical2, Numerical3, Numerical4, River-bank1;
    EFFECTS: (ON Two-People-Boat River-bank1)
             (PRODUCE Cannibals River-bank1 1)
             (PRODUCE Missionaries River-bank1 1)
             (STATE Numerical5 Numerical6 Numerical7 Numerical8
                    River-bank1);
END PLOT END OPERATOR


OPERATOR: MOVE-2-C
ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1,
                 Numerical5 is (SIPE+ -2 Numerical1),
                 Numerical6 is  (SIPE+ 2 Numerical3),
                 Numerical7 is (SIPE+ 0  Numerical2),
                 Numerical8 is  (SIPE+ 0  Numerical4), River-bank2 ;
PRECONDITION: (OPPOSITE-BANK River-bank1 River-bank2)
                 (NOT (STATE Numerical6 Numerical8 Numerical5 Numerical7
                              River-bank1))
                 (LEVEL>= Cannibals River-bank2 2)
                 (OR (LEVEL Missionaries River-bank1 0)
                     (LEVEL>=  Missionaries River-bank1 Numerical6));
PLOT:
  PROCESS
    ACTION: ROW-2-C
    ARGUMENTS:  Numerical1, Numerical2, Numerical3, Numerical4, River-bank1;
    EFFECTS: (ON Two-People-Boat River-bank1)
             (PRODUCE Cannibals River-bank1 2)
             (STATE Numerical6 Numerical8 Numerical5 Numerical7 River-bank1);
END PLOT END OPERATOR


OPERATOR: MOVE-1-C
ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1,
                 Numerical5 is (SIPE+ -1 Numerical1),
                 Numerical6 is  (SIPE+ 1 Numerical3),
                 Numerical7 is (SIPE+ 0  Numerical2),
                 Numerical8 is  (SIPE+ 0  Numerical4), River-bank2;
PRECONDITION: (OPPOSITE-BANK River-bank1 River-bank2)
                 (NOT (STATE Numerical6 Numerical8 Numerical5 Numerical7
                              River-bank1))
                 (LEVEL>= Cannibals River-bank2 1)
                 (OR (LEVEL Missionaries River-bank1 0)
                     (LEVEL>=  Missionaries River-bank1 Numerical6));
```

```
PLOT:
  PROCESS
    ACTION: ROW-1-C
    ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1;
    EFFECTS: (ON Two-People-Boat River-bank1)
             (PRODUCE Cannibals River-bank1 1)
             (STATE Numerical6 Numerical8 Numerical5 Numerical7 River-bank1);
END PLOT END OPERATOR


OPERATOR: MOVE-2-M
ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1,
           Numerical5 is (SIPE+ -2 Numerical2),
           Numerical6 is  (SIPE+ 2 Numerical4),
           Numerical7 is  (SIPE+ 0 Numerical1),
           Numerical8 is (SIPE+ 0 Numerical3), River-bank2 ;
PRECONDITION: (OPPOSITE-BANK River-bank1 River-bank2)
              (NOT (STATE Numerical8 Numerical6 Numerical7 Numerical5
                          River-bank1))
              (LEVEL>= Missionaries River-bank2 2)
              (OR (LEVEL Missionaries River-bank2 2)
                  (LEVEL<=  Cannibals River-bank2 Numerical5)),
              (LEVEL<=  Cannibals River-bank1 Numerical6);
PLOT:
  PROCESS
    ACTION: ROW-2-M
    ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1;
    EFFECTS: (ON Two-People-Boat River-bank1)
             (PRODUCE Missionaries River-bank1 2)
             (STATE Numerical8 Numerical6 Numerical7 Numerical5 River-bank1);
END PLOT END OPERATOR


OPERATOR: MOVE-1-M
ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1,
           Numerical5 is (SIPE+ -1 Numerical2),
           Numerical6 is  (SIPE+ 1 Numerical4),
           Numerical7 is  (SIPE+ 0 Numerical1),
           Numerical8 is (SIPE+ 0 Numerical3), River-bank2;
PRECONDITION: (OPPOSITE-BANK River-bank1 River-bank2)
              (NOT (STATE Numerical8 Numerical6 Numerical7 Numerical5
                          River-bank1))
              (LEVEL>= Missionaries River-bank2 1)
              (OR (LEVEL Missionaries River-bank2 1)
                  (LEVEL<=  Cannibals River-bank2 Numerical5)),
              (LEVEL<=  Cannibals River-bank1 Numerical6);
PLOT:
  PROCESS
    ACTION: ROW-1-M
    ARGUMENTS: Numerical1, Numerical2, Numerical3, Numerical4, River-bank1;
    EFFECTS: (ON Two-People-Boat River-bank1)
             (PRODUCE Missionaries River-bank1 1)
             (STATE Numerical8 Numerical6 Numerical7 Numerical5 River-bank1);
END PLOT END OPERATOR

;;===================== Causal Theory =========================

CAUSAL-RULE:  On-Bank-Not-On-Other-Bank
```

```
ARGUMENTS: Boat1, River-bank1, River-bank2;
TRIGGER: (ON Boat1 River-bank1);
PRECONDITION: (ON Boat1 River-bank2) (OPPOSITE-BANK River-bank1 River-bank2);
EFFECTS: (NOT (ON Boat1 River-bank2));
END CAUSAL-RULE

CAUSAL-RULE: Produce-and-Consume-Cannibals
ARGUMENTS: Cannibals, River-bank1, Numerical1, River-bank2;
TRIGGER: (PRODUCE Cannibals River-bank1 Numerical1);
PRECONDITION: (OPPOSITE-BANK River-bank1 River-bank2)
EFFECTS: (CONSUME Cannibals River-bank2 Numerical1);
END CAUSAL-RULE

CAUSAL-RULE: Produce-And-Consume-Missionaries
ARGUMENTS: Missionaries, River-bank1, Numerical1, River-bank2;
TRIGGER: (PRODUCE Missionaries River-bank1 Numerical1);
PRECONDITION: (OPPOSITE-BANK River-bank1 River-bank2)
EFFECTS: (CONSUME Missionaries River-bank2 Numerical1);
END CAUSAL-RULE

STOP

;;======================= Problem ========================

(DEFINE.PROBLEM)

PROBLEM: MC-Problem
    GOAL: (MOVE-EVERYBODY)
END PROBLEM

STOP
```

☆U.S. GOVERNMENT PRINTING OFFICE: 1994-610-126-50084

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.